PROJECT

DATA MINING AND LEARNING ALGORITHMS

Parts: 1) California Energy Dataset
2) Amazon Product Reviews

Part 1

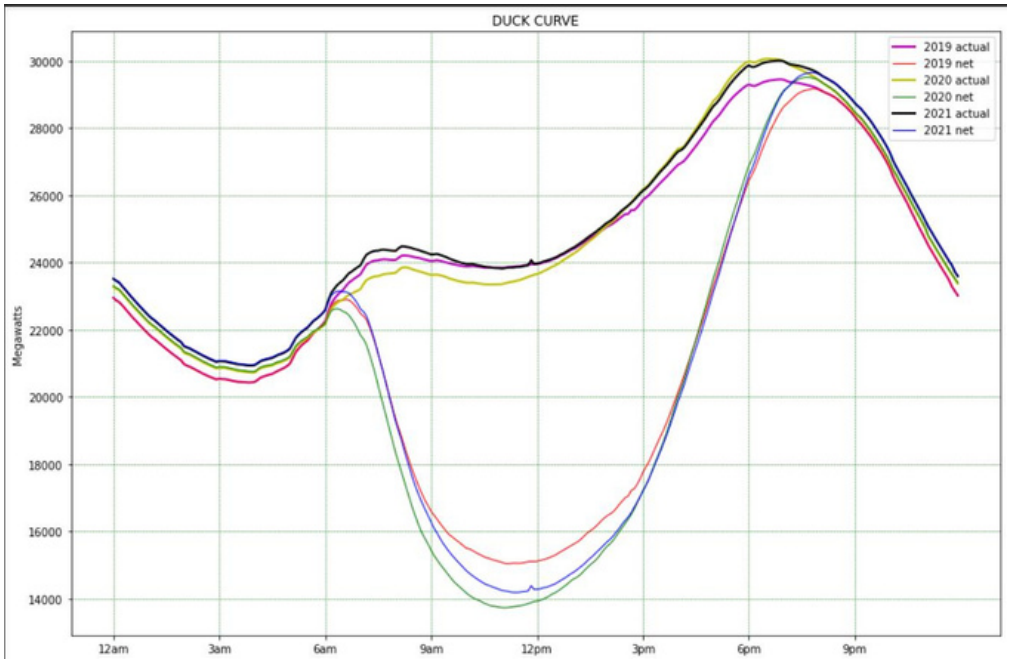
Introduction:

For the work we will use python language along with NumPy libraries,
pandas, matplotlib, sklearn, TensorFlow. The implementation will take place in Google's Colab environment and the
data has been uploaded to the drive from where it is retrieved every time we run it program.

Question 1.A) Here we need to group our data into a csv file for demand
and production. We will save and upload the two resulting csv files
those in the drive because the rest of the queries take these 2 as input.
In the sources we notice that for some days the last rows with data (max 10) are missing
so we fill these values with the last available value without losing any important
information so that all files have 288 lines (1 day).
Also for the demand we delete the last line as it corresponds to the beginning of the next day of the day and fill in any blanks with 0.

To visualize the data we will present the duck curve which shows the
problem of renewables compared to demand
For graphic purposes we will only consider solar energy and calculate the average
serum every year for the whole day.

Duck Curve



In the above graph we observe the real demand and the net demand, i.e. demand less photovoltaic generation. The diagram shows that during the day solar energy covers the largest amount of demand while the night hours almost zero.

It is also observed that every year the solar energy produces more and more energy -> h the duck's belly grows.

The code is here:

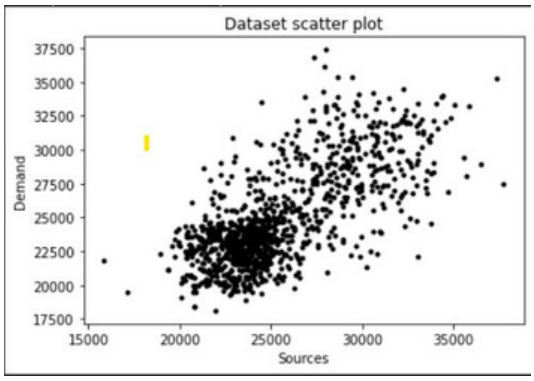
Question 1.B)

In this query we need to group the energy values of each day in order to identify outliers.

So we will use the DBSCAN algorithm because it calculates outliers.

First we will load our consolidated data from the 1st query and after summing them all the prices of the sources we will find the average price of demand - supply for each day.

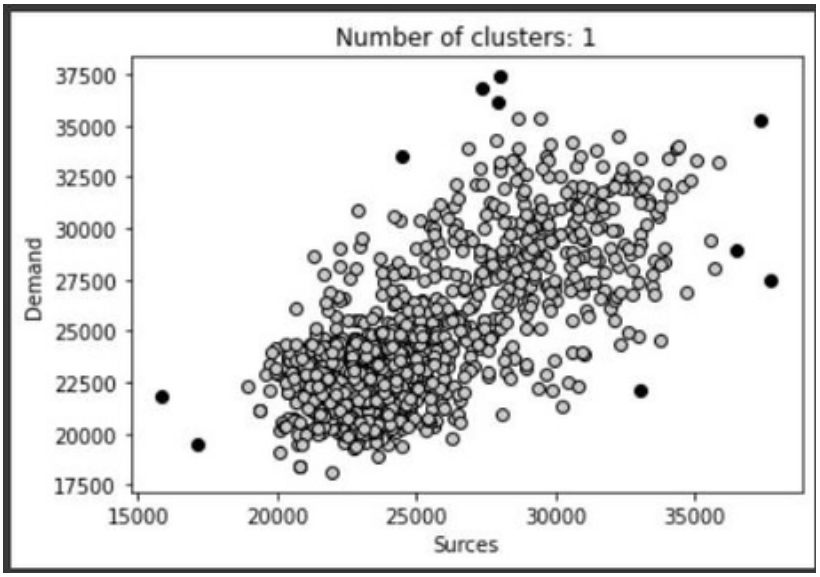
Initial data



We will implement DBSCAN with parameters eps=2000, min_samples=18

And in the results it produces 1 cluster with 10 outliers.

The clustered data



The black dots are the days-outliers and as we can see they are peripheral to them of dots where they stand out.

Finally we find the dates to which the outliers correspond

The following dates where estimated as outliers:

	date	supply	demand
0	2019-02-12	33068.97	22141.15
1	2020-04-18	27938.97	36145.12
2	2020-04-19	37751.61	27470.74
3	2020-04-20	27356.92	36866.26
4	2020-04-21	27992.07	37383.51
5	2020-04-23	36494.59	28891.85
6	2020-04-28	37356.15	35267.95
7	2021-05-07	24453.81	33521.30
8	2021-09-12	15856.02	21812.47
9	2021-12-10	17182.57	19514.36

The code is here

Question 1.C)

Here we are asked to predict the energy production from non-renewable sources for each time of day using an LSTM regression.

DATA PRE-PROCESSING

Initially we consider coal and natural gas as non-renewable sources, but because coal has an average production price of 14.24 MW while natural gas has an average price of 8286.76

MW we add them and make a new column with "fossil" which contains the sum of the 2 sources.

Our data will have the form: X-(m_samples, x_time_steps, n_features), Y-(m_samples, y_time_steps,1).

The number of time_steps in Y is determined by how many times of the day we want to predict and based on this we also choose the number of time_steps in X, which must

is equal to or greater than Since we want to predict the value for the next moment in time we will have: Y(~300k, 1).

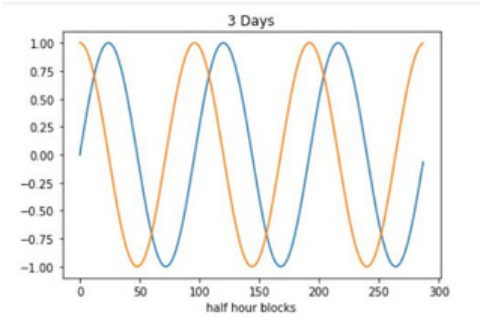
FEATURE SELECTION
First, since we have 17 features, we will make the autocorrelation table using it spearman distance as it works better with non-linear correlations.
The features that have a value in the table less than 0.1 will be removed without we lose information.

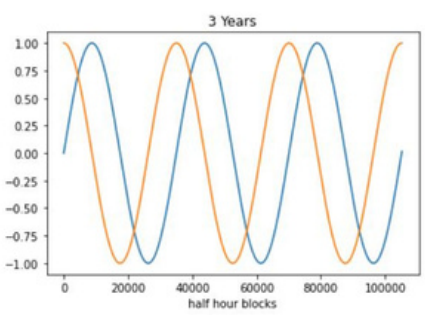
	current demand	solar	wind	geothermal	biomass	biogas	small hydro	nuclear	large hydro	batteries	imports	other	fossil
current demand	1.000000	0.147587	0.061165	-0.048298	0.250501	0.029666	0.242581	0.042514	0.362011	0.158429	0.156059	0.028444	0.447895
solar	0.147587	1.000000	-0.138829	-0.055759	0.099024	-0.070244	0.082094	0.120849	-0.103830	-0.167431	-0.707638	0.012360	-0.315993
wind	0.061165	-0.138829	1.000000	0.043016	0.043972	0.036827	0.149460	0.217448	0.159707	0.059386	-0.014494	0.008767	-0.222815
geothermal	-0.048298	-0.055759	0.043016	1.000000	-0.097802	0.118300	-0.053426	-0.223412	-0.038145	0.002596	0.022427	0.026050	0.002228
biomass	0.250501	0.099024	0.043972	-0.097802	1.000000	0.263432	0.413199	0.221754	0.291362	0.045666	0.111193	0.040624	-0.057897
biogas	0.029666	-0.070244	0.036827	0.118300	0.263432	1.000000	0.300833	-0.011155	0.233637	0.034957	0.069800	0.051587	-0.024662
small hydro	0.242581	0.082094	0.149460	-0.053426	0.413199	0.300833	1.000000	0.245742	0.803089	0.182972	0.056956	0.036440	-0.123325
nuclear	0.042514	0.120849	0.217448	-0.223412	0.221754	-0.011155	0.245742	1.000000	0.182258	-0.007877	-0.045081	0.000508	-0.302565
large hydro	0.362011	-0.103830	0.159707	-0.038145	0.291362	0.233637	0.803089	0.182258	1.000000	0.306560	0.219887	0.031243	0.130438
batteries	0.158429	-0.167431	0.059386	0.002596	0.045666	0.034957	0.182972	-0.007877	0.306560	1.000000	0.172218	0.002135	0.146350
imports	0.156059	-0.707638	-0.014494	0.022427	0.111193	0.069800	0.056956	-0.045081	0.219887	0.172218	1.000000	0.007474	0.357766
other	0.028444	0.012360	0.008767	0.026050	0.040624	0.051587	0.036440	0.000508	0.031243	0.002135	0.007474	1.000000	-0.009036
fossil	0.447895	-0.315993	-0.222815	0.002228	-0.057897	-0.024662	-0.123325	-0.302565	0.130438	0.146350	0.357766	-0.009036	1.000000

We notice that the sources: geothermal, biomass, biogas and other have almost zero correlation

which is logical because these sources produce much less energy but are also stable in permanent level. We also see that demand has the highest correlation with 0.44

TIME DIMENSION





As an input to our network we will have all the sources as well as the demand for each moment plus a time parameter that we will mention below and to output the fossil column in the next ones moments.

Production from non-renewable energy sources depends mainly on demand at any given time. Demand follows 2 patterns, one for the length of the day and one for its length
of time. That is, it plays an important role if we predict days in January compared to days in May and for which moments of the day we predict and we do not have this pattern in the data.
For this we will take the index of our data and apply a sin and cos transformation with period 1 day and 1 year.
We use sin and cos because we want each point to have a unique value and for each period the
semitropic functions have 2 eigenvalues.
This gives a unique value for each time of day and each day in time.

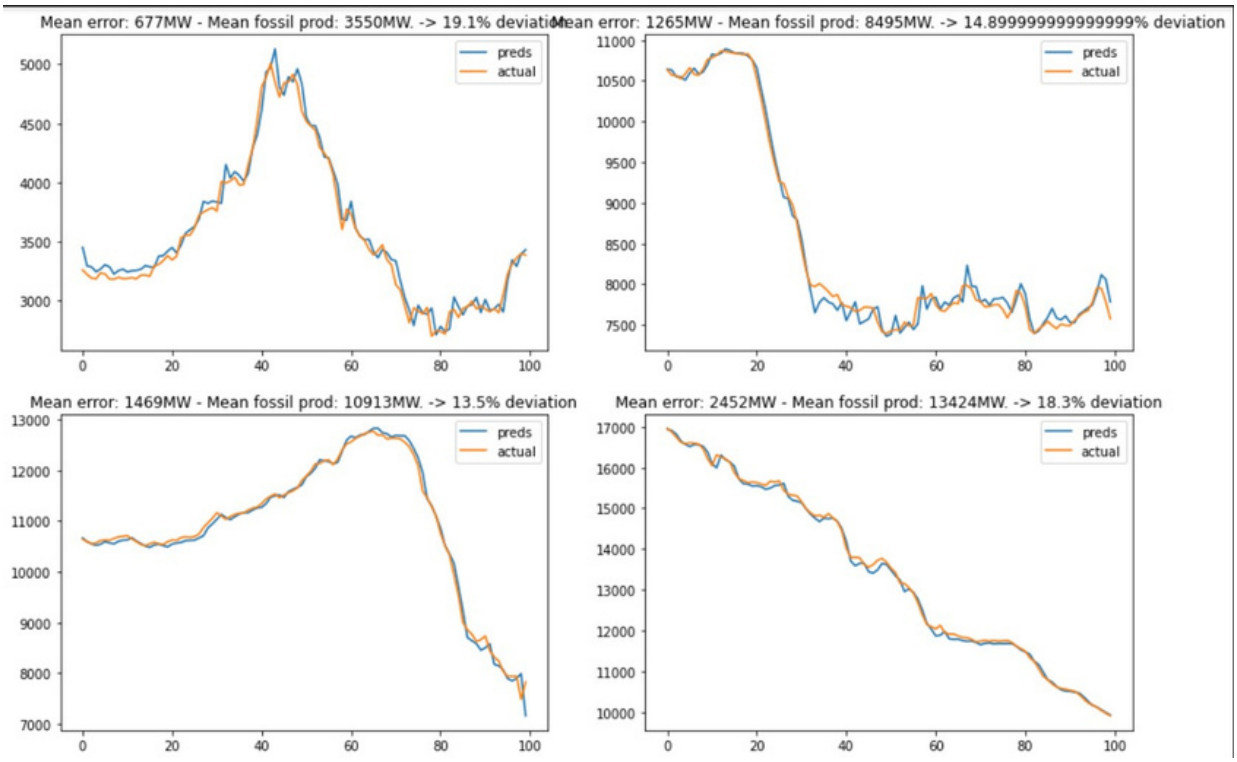
NORMALIZATION

Since the data values range from -30 to +70k we will normalize the data
us with the transformation: Z = (X – mean) / Std for each column keeping the 2 variables for
our target column to reverse the transformation after predictions.

MODEL SELECTION

The model we will use consists of 4 layers
LSTM -> Dropout -> Dense -> Output.
For loss we use Mean Squared Error while for metric RMSE and for optimizer Adam.
To train our model we use EarlyStopping in order to keep them
better results.
Within a few epochs the model approaches RMSE = 0.0498 on the validation dataset.
While in testing
RMSE = 0.0455

4 random 100-moment time periods from the test dataset.



From the results we get we have that

```
The mean error is:  178.75  Megawatts
The mean fossil fuel energy productions is:  7873.18  Megawatts
The standard dev. fossil fuel energy productions is:  3018.9  Megawatts
Mean deviation percentage:  2.27 %
```

The average value of the error in MW values is 178MW, this is 2% of the standard deviation that fossil fuel power plants have a value that is quite acceptable.

The code is here:

Part 2

Question 2)

Here we are asked based on the comments that Amazon customers have made to predict the rating they have put on this product using

a Random Forest.

The scores range from 1 to 5 so we have 5 hells.

We will convert the comments into vectors using the word2vec model

of Google which however applies to individual words so we will

calculate the average for the entire comment.

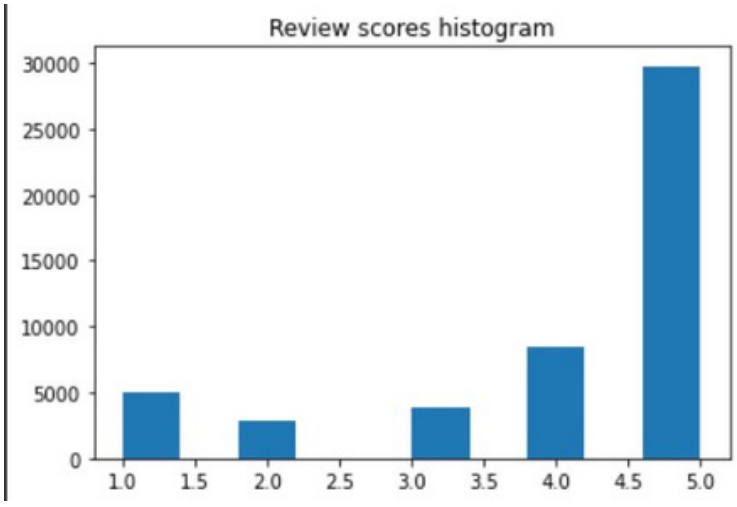
First we look at our data structure

	Text	Score
0	The description and photo on this product need...	3
1	This was a great book!!!! It is well thought t...	5
2	I am a first year teacher, teaching 5th grade....	5
3	I got the book at my bookfair at school lookin...	5
4	Hi! I'm Martine Redman and I created this puzz...	5

In which as we noticed there are some double entries ~700 of them

which we remove.

The breakdown is:



We see that they follow the J-distribution like almost all datasets with product reviews.
Our data is very unbalanced with ~30k for class 5 and ~3k for
class 2 so we have to balance them "somewhere in the middle" in order for each
class to have a similar number of samples to make the training better.
First we divide the data into train – test as it does not affect us if in testing
the data is unbalanced.
In the train data we have the following sets of classes.

So we choose to resample all hells to 5000 samples.
This means that for classes 4.5 we will discard samples while for the rest
we will use some samples and 2nd time.
Specifically for class 2 we will more than double its number as it is a lot
small in convergence with the rest.
In total we will have 25000 training samples.

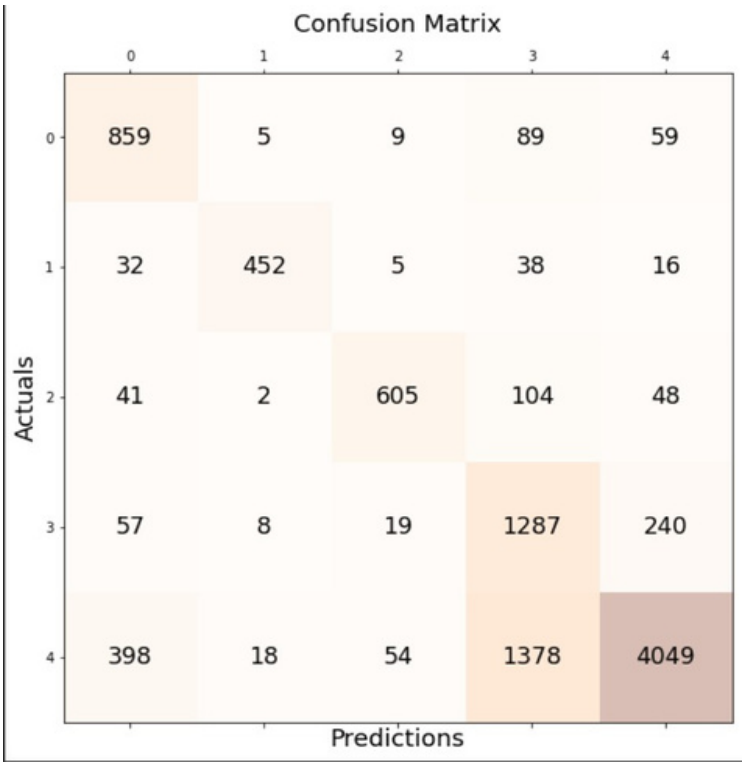
To convert the texts into vectors we follow 2 steps:
- In each comment we remove punctuation, lowercase the letters and break the words into a list.
- We make each word a vector with word2vex and at the end we calculate
the average.
Our data has dimension (n_samples, 300).

```
a5 = train_df[train_df['Score']==5].count()[0]
a4 = train_df[train_df['Score']==4].count()[0]
a3 = train_df[train_df['Score']==3].count()[0]
a2 = train_df[train_df['Score']==2].count()[0]
a1 = train_df[train_df['Score']==1].count()[0]

print(a1, a2, a3, a4, a5)
```

```
4025 2294 3085 6567 23514
```

In the Random Forest Classifier we select 400 trees and as a classification criterion the entropy and the train measure we get the results.
In the test data we make the confusion matrix



We also calculate the precision/recall/f1-score

The model shows 73.5% accuracy and the f1 score ranges from 0.571 to 0.879 for the variables.

	precision	recall	f1-score	support
1	0.619	0.841	0.713	1021
2	0.932	0.832	0.879	543
3	0.874	0.756	0.811	800
4	0.444	0.799	0.571	1611
5	0.918	0.687	0.786	5897
accuracy			0.735	9872
macro avg	0.758	0.783	0.752	9872
weighted avg	0.807	0.735	0.750	9872

We also notice in class 5 that several predictions are mapped to 4, logically the comments are quite similar but quite strangely they are mapped to class 1

The code is here